Exception detection and handling in mission control for mobile robots

Andersen, Thomas Timm; Andersen, Nils Axel; Ravn, Ole

Published in:
Proceedings of 8th IFAC Symposium on Intelligent Autonomous Vehicles

Link to article, DOI:
10.3182/20130626-3-AU-2035.00060

Publication date:
2013

Exception detection and handling in mission control for mobile robots

Thomas Timm Andersen*, Nils Axel Andersen*
and Ole Ravn*

**Department of Electrical Engineering, Technical University of Denmark, Lyngby, Denmark*
(e-mails: ttan@elektro.dtu.dk, naa@elektro.dtu.dk, or@elektro.dtu.dk).

Abstract: This paper introduces a method for robust, rule-based mission control for mobile robots in a modular framework. Due to the modularity of the framework, it is possible to use both hierarchical control and reactive behavior seamlessly to find solutions to both planned and unplanned event in the mission execution.

A demonstration example for office navigation is presented along with considerations for rules that should ensure robust solving of missions.

Keywords: Expert systems, Autonomous mobile robots, Hierarchical systems, Robust performance, Robot Navigation, Robot programming

1. INTRODUCTION

For decades, autonomous mobile robots have been expected to have a major impact on tomorrow's society, with journalist often proclaiming that the robots are coming to take over the world. Yet, the robots that have actually left the labs are neither of a number nor with a functionality anyway near of what could have been expected, let alone taking over anything.
One of the major showstoppers in preventing autonomous mobile robots from making it out of the lab is the robustness of one of the most elemental skills needed; mission control. This is particularly apparent in navigation - when the robot loses its heading, gets lost, or stuck, it is impossible for the robot to solve a mission that is related to moving to a specific position. While research in different approaches to navigation are progressing fast, the algorithms usually fall short when applied to a real robot in the dynamic and non-deterministic human world outside of a lab's static or controlled environment.

In general, modern sensor technology have aided researches in coming up with some impressive methods, but it is not likely that no missed detections or any false positive detections can be removed completely. Other problems can also disrupt mission solving, from obstructed wheels to crashed software.

1.1 Previous work

A recent survey of fault diagnosis and fault tolerant control for wheeled mobile robots is given in (Duan, et al., 2005). Several methods are analysed to conclude on the most common problems, and some future trends within fault diagnostics and handling, including integrating models, control, and knowledge in a uniform framework, is presented. (Marco, et al., 1996) provides an experimental implementation of a hybrid (mixed discrete state/ continuous state) controller for autonomous underwater vehicles, using rules for strategic mission control and hard real time for motion control at an executive level. Others, like (Zhou, et al., 2005) have also shown that using an expert system for mission control can be advantegues.

The Mobotware framework introduced at IAV2010 (Beck, et al., 2010) presents a modular, socket-based framework, where plugins can be used to extend the robots capabilities, both in real-time and non real-time control.

The knowledge gained from these contributions have been used to develop a solution based on a rule-based expert system, that is able to handle all parts of mission control for a mobile robot, and does so in a modular framework.

When analysing the sensor input and the perceived output of the individual modules, the situations where a navigation algorithm falls short can be detected and thus give the opportunity to degrade to another method gracefully by handling the exception and thus saving the mission. When the system can detect exceptions and handle them, algorithms that are known to be effective, but error-prone in some situations, can be used with great benefits, as long as an opposing behaviour to the error-related situation can be defined.

The logic behind this is that while it is very difficult in an algorithm to detect if something is logically wrong with the input or result, i.e. planning a route beneath an object perceived to be levitating, usually even an untrained human can easily see if the robot is behaving less than optimal. By formulating the reasoning that humans do to detect this into some rules and using it in an expert system to do sanity checks on input, planned output, and the current state of the

modules, all together at once, a much more robust behaviour can be obtained.

What exceptions to look out for, and appropriate handling of these, is highly application specific. This paper will therefore, after introducing the framework for using expert systems in mission control and sketching an example scenario, give some thoughts for considerations that should be applied when designing robust mission control. The paper will focus on navigation related problems, but the method and considerations described are applicable to all problems related to autonomous mobile robots.

2. SYSTEM ARCHITECTURE

The Mobotware framework is used for simulation and controlling of robots, while Jess (Friedman-Hill, 1998) is used as the expert system doing the exception detection and handling. As argued in (Arkin, 1989), the use of a hierarchical control method like Mobotware and a reactive control method like Jess can yield robust, flexible, and generalizable navigation. The two are tied together using a Jess package named JessMW, which is introduced in section 2.3.

2.1 Mobotware

The Mobotware framework has three core modules:

- *Robot Hardware Daemon (RHD)* Flexible hardware abstraction layer for real-time critical sensors
- *Mobile Robot Controller (MRC)* real-time closed-loop control of robot motion and mission execution.
- *Automation Robot Servers (AURS)* Advanced framework for processing complex sensors and non real-time mission planning.

These modules allow for a two-dimensional decomposition: temporal and functional.

The temporal dimension divides Mobotware into a hard and a soft real-time constrained section, where RHD can handle the sensors and actuators requiring strict timing like wheel encoders and motors, making MRC able to run the robot at a desired speed for a given distance, while AURS handles the non timing-critical sensors like cameras and processes the data from these sensors to extract information about the sourrounding environment.

The functional decomposition divides the framework in levels of increasing abstraction from the hardware abstraction layer in RHD, to reactive execution in MRC up to perception and planning with AURS.

The modular architecture is further strengthened by the use of plugins to implement both sensor interfaces and data processing algorithms, thus making it possible to use i.e. different methods for navigation, without altering anything else in the system.

All the core modules in Mobotware are connected through low latency TCP/IP connections, making it possible to easily exchange information both between the modules as well as with external processes, and distribute the computations across several computer platforms.

2.2 Jess

The Java expert system shell is an expert system implemented in Java that processes a CLIPS-like rule-based language. It can do both forward and backward chaining of rules, and using its RETE network it can handle several 100.000s of rules per second on a modern computer, while maintaining a huge fact base (Friedman-Hill, 1998).

As Jess is implemented in Java, it has many object-oriented features including a direct interface to Java components. This makes it possible, via Java libraries, to connect to other processes on the computer and create shadow facts that

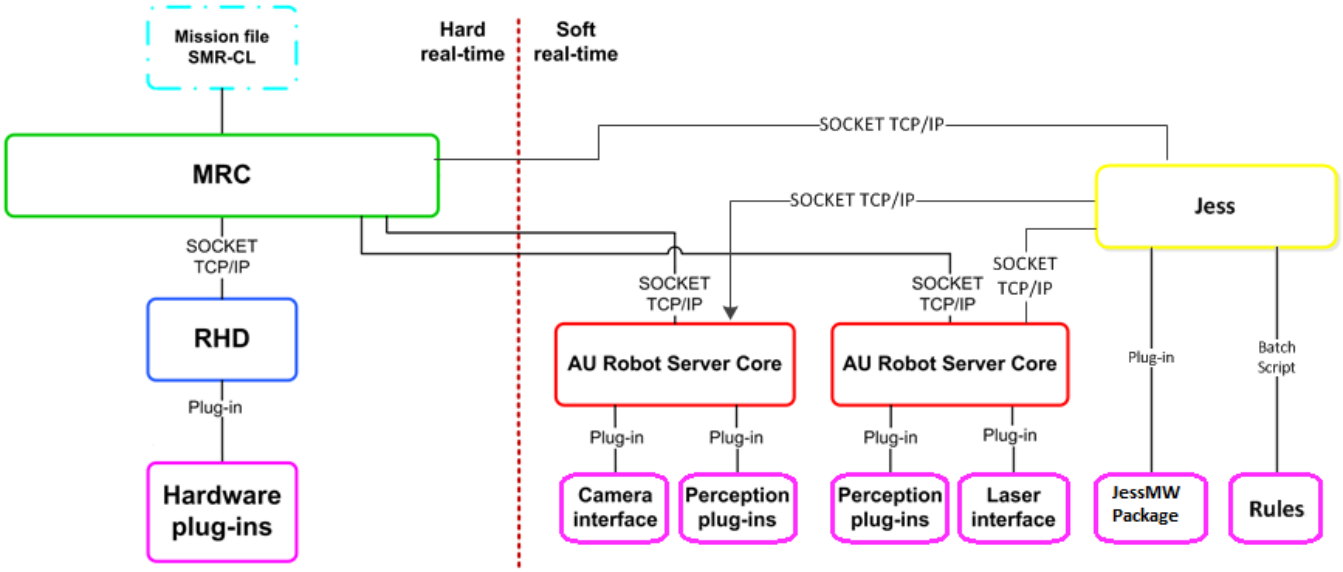

Fig. 1. Overview of the Jess-Mobotware framework

represent the knowledge obtained from the other processes. This ability has made Jess widely popular in as diverse fields as mobile robotics (Hladek, et al., 2009), web services (Grove, 2000), fuzzy logic (Orchard, 2001), and diagnostics and learning (Ong, et al., 2001).

2.3 JessMW

In order to bridge the gap between Jess and Mobotware, the Jess package JessMW is made available. Using Java, the functionality of Jess is expanded by making it possible to communicate with MRC and AURS using the TCP/IP connections. Using these connections, the data concerning the surrounding environment perceived by AURS can be pushed to Jess and turned into facts about the perceived world, known as shadow facts. Using rules, Jess can then compare the perceived information with any prior or learnt knowledge of the environment and thus perform a sanity check on the information before the robot tries to act on it. Not only can this be used to prevent actions on detections known not to be possible, but it can also be used both for making the robot branch into a searching behaviour, if something that according to the internal state should be found but is not, and for challenging the robot's beliefs about its current status, which might require the robot to go back in order to reassess the current situation of itself and the environment.

Likewise, information regarding the robot's status, like current motion instruction and odometry information, is fetched from MRC and turned into shadow facts by JessMW. It is also possible to get information from the time-critical sensors using this connection, and as Jess fetches this information via MRC and updates its own local knowledge base, it does not need to lock sensor values during search operations. This ensures that there are no risks of violating any real-time constrains. This is an issue sometimes observed when combining expert systems and real-time critical systems, and is part of the motivation for using a modular approach.
Via the connection to MRC, Jess can also send, stop and flush current motion instructions which make it possible not only to detect exceptions in mission execution, but also to handle them by imposing another solution to the current mission (Jørgensen, et al., 2008). It can also monitor the execution of said motion instructions and detect i.e. if the low level security function in MRC has suspended motion due to a blocking obstacle in front of the robot.

Finally, JessMW can communicate with several Mobotware-enabled robots simultaneously, making it seamless to exchange information about the environment between robots and help each other to detect any exceptions or abnormalities.

An overview of the framework can be seen in Fig. 1.

3. IMPLEMENTATION

To verify the proposed solution and the system architecture, extensive testing has been carried out, both in the methodological demonstration example described below as well as by several student at Automation and Control, DTU Electrical Engineering, who without prior exposure to robotics successfully used the principles to solve missions where cooperation between robots in accessing the state of the environment was a key element.

3.1 Demonstration scenario

The proposed solution was tested on a small mobile robot (SMR) running Mobotware. The SMR is a differential driven robot with wheel encoders for odometry, 1D IR distance sensor for low level collision prevention, laser scanner, and camera.

The robot was tasked to navigate through an office environment, which is controlled enough to ensure repeatability, but dynamic enough, due to human presence, to ensure problems for most navigation algorithms. Moving furniture in front of the robot or additional traffic can also easily be introduced to challenge the robot further. A partial map of the office space can be seen in Fig 2.

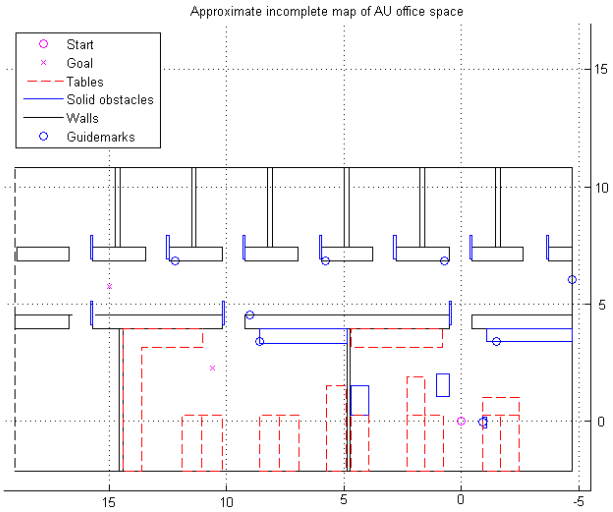

Fig. 2. Approximate map of office space

Even the optimal solution should require a travelled distance and enough turns to ensure the mission can't be solved robustly with odometry alone. Also, two different goals were defined and specified at runtime, with multiple successive runs to each goal.

3.2 Algorithms for navigation

For global path planning, a graph node planner is used to calculate the most feasible path towards the goal. The expert system controls the path planner's nodes, making sure the knowledge of all landmarks' positions is known by the path planner.

For global localization, landmark-based navigation is used. This approach has the advantage of only requiring a minimum of prior knowledge about the surrounding environment, namely locations of landmarks and traversable routes between them, compared to other global map-based methods.
It is implemented using 2D barcodes, easily detectable by the

on-board camera. With proper calibration and optimal detection positioning (~1 m distance, 10° angle), position and heading estimation relative to the barcode can be obtained to within ±2 cm and ±1° (Andersen, 2006). As the barcode is printed on paper it is only detectable from one side.

For local path planning and obstacle detection, a method based on jumps and openings in a laser range scanner image is used (Understrup, 2011). It evaluates the environment in front of the robot and tries to move towards a given goal position. It is very opportunistic in that it plans a path and then executes it, without evaluating if something suddenly blocks the path. This is left to the low level collision avoidance to detect.

For local localization, odometry is used. To prevent the position error from growing unbounded, the odometry information is zeroed at each barcode detection and thus relating the local and global position.

All the used navigation algorithms are existing parts of the Mobotware framework.

4. BEHAVIOURS

To control the mission progression, a set of goal-seeking rules, or behaviours, are defined in the expert system. These can be divided into three groups: Input sanity validation, mission execution and exception handling. As stated earlier, the appropriate action for all the behaviours is a design choice, based on the specific application as well as the dependability of each sensor and algorithm.

4.1 Input sanity validation

This group of rules verifies all input related to mission progression. In the test example described in this paper, this is primarily related to the detection of landmarks.

If a landmark is reported spotted by the robot, the system should verify that the landmark is logically observable from the robots current position, i.e. that the robot is not currently behind the barcode, the relative orientation is unrealistic or the distance is further than what is known to be reliable for the detection algorithm. If the input is validated, the robot's position is updated based on this.
But if the input is invalidated, the design choice of what action to take is based on what sensor output (odometry vs. landmark detection) is most reliable for the current robotic system.
One approach, and the one implemented in the demonstration, is to reject such detections, assuming they are reflections or simply false positives from the sensor.

Another issue is what to do if the system detects a landmark that it has no prior knowledge about. For a system operating in an assumed known environment, it might be most reasonable to ignore the detection, while a system designed for exploration probably would add the landmarks position, maybe flagging it unreliable until it is independently validated several times.

4.2 Mission execution

These rules govern mission execution in general, including planning, monitoring of progression, and issuing of motion instructions.

Using the path planner, the robot moves towards the goal position using the known landmarks en-route. Using odometry, the robot tries to position itself optimally accordingly to the barcode, so as to get the best estimate of position and heading. This is found to be at a small angle (~10°) relative to the barcode.

The locations visited, the current target location as well as planned locations to visit is all kept in the expert system's working memory.

4.3 Exception handling

These are the rules that have the ability to add real robustness to mission solving. The behaviours described above assume some flow of events in a predictable order and that actions will lead to reactions, i.e. a motion instruction will lead to a corresponding movement or a landmark is visible and thus detected at a certain position.

But if this is not the case, behaviours should be defined that can detect the abnormalities, and then handle them in order to save the mission. This is also the major difference compared to input sanity validation, as an invalidated input reading should not jeopardise the mission, as long as a valid reading can be obtained afterwards without any active handling.

Obviously it is difficult for an algorithm, trying to detect something, to figure out whether it failed because nothing is there, or because something else is wrong.
This is where an expert system is critical, as it has the opportunity to diagnose or try redundant systems to detect what is wrong. But for the expert system to have any chance at this, it is necessary to have a broad understanding of what can lead to the used algorithms failing.

In the end, the appropriate action is highly dependent on the system's properties and the nature of the mission. If the detection algorithm is weak but localization is strong, it might be best to drive on, hoping to see the next landmark on the route. If both are equally weak (or strong), it would probably be better to try and search for the landmark before giving up, and if it is the localization that is the weak part, maybe it would be better to go back the same route to a previous landmark and get a new position estimate to zero the local localization error. Or maybe a human operator should be notified, if the mission depends on the robot finding the landmark or it has been undetectable several times. In the test example, a searching behaviour is initiated.

The same goes for motion instructions; if something blocks the robot and thus prevents motion, a mission execution scheme that waits for the robot to finish its movement will stall if no thoughts are given to exception handling.

In the test example, an evasive behaviour is triggered if the low-level collision avoidance halts the robot, making it reverse and try to find another local path around the object. If that does not work, it will head back to the previous landmark to get a new orientation fix.

In general, when drawing a flow chart or state diagram of the expected mission progression, each transition should be analysed to map what part that might fail and thus prevent the transition. Likewise, each state should be analysed to consider what will happen if none of the expected transitions occur. Will the robot stall, will it keep moving in a possible wrong direction or could it might somehow end up in an undesirable state.

It is also worth to consider that the exception handling might also fail, and thus further degrading might be necessary. How redundant a system should be in the end is a design question that usually depends on a trade-off between desired robustness and resources, both time-wise, money-wise and computational power.

5. RESULTS

In the presented test case, the robot successfully completed its mission in all ten trials.

To show the robustness of the solution, different things were done to try and disrupt the mission control.

Fig. 3 shows a map of a test run where the robot was placed at another orientation than what the internal state of the robot represented.

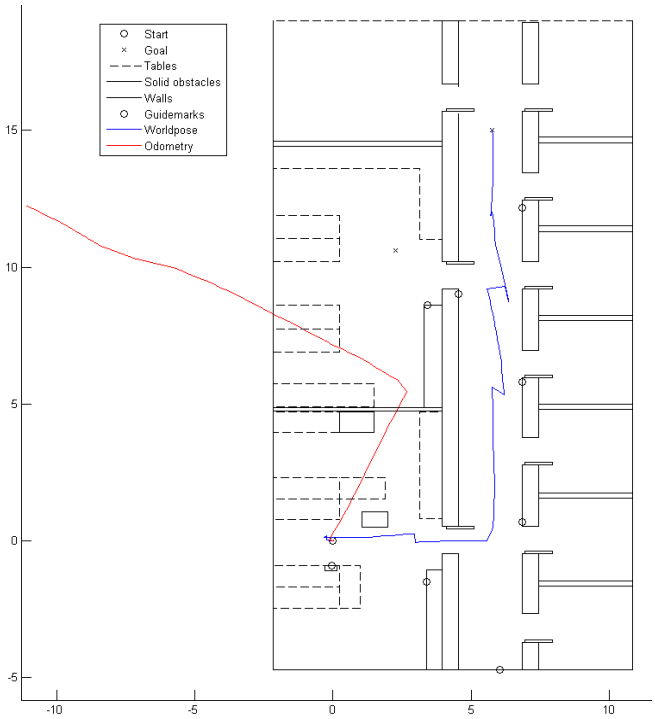

Fig. 3. Map plot of a demonstration test run.

The red line represents the robot's own internal belief of where in the world it thinks it is, whereas the blue shows where the expert system's thinks the robot is, based on the sensed information regarding the environment. Thus the sudden jerks that only occur on the *worldpose* line happen when a landmark is detected.

This ability is similar to what other global localization methods can achieve, but goes to show that localization can be done in expert systems, and familiarizes the reader with the map structure used in the following. It should be noted that the overlaid map does not necessarily shows the true

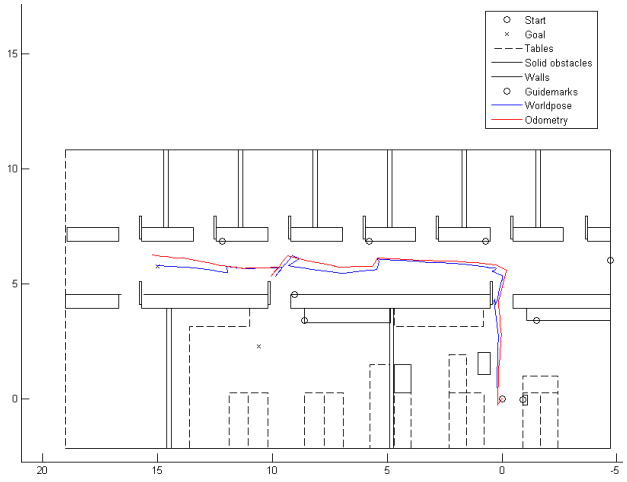

Fig. 4. Path suddenly blocked by an opening door

position of the robot, but is only added to give the reader a sense of where the robot is in the environment. Due to layout constrains in this paper, the figure has been rotated.

In Fig 4 the robots path is suddenly blocked by an opening door. To avoid collision, the robot stops due to the low level collision avoidance. It then reverses and find a new path to the target.

The ability to detect and handle a security-related suspension of the execution was tested in several of the trials to prove that the robot could cope with a changing environment where humans move about and might unintentionally place objects in front of the robot.

Fig 5 shows the robot trying to find a landmark at an expected position. When this is unsuccessful, the robot starts searching for the landmark in a predefined pattern. This show an example of handling a sensor or detection algorithm that fails to deliver an expected result. This is a surprisingly common problem when moving a robot system from a simulated world to a real world implementation. The causes and consequences of such an event, or the opposite with a false positive detection, should be analysed with both great care and conservatism to increase the robustness of a mission controller, along with workarounds to the problem that works independent of the sensor or algorithm that is failing.

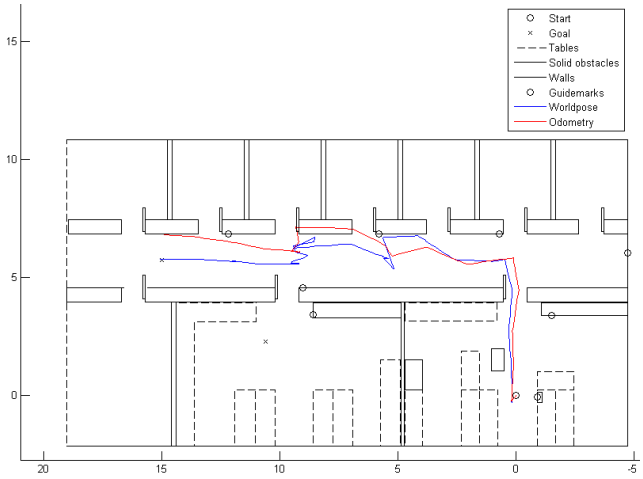

Fig 5. The robot searches for the landmarks

6. CONCLUSIONS

In this paper design for robust mission control using an expert system has been discussed. The field is highly application specific, so defining generic rules or a standard for designing robust mission control is not possible. Nevertheless, if the reliability of autonomous mobile robots is to gain enough trust to one day move out of the laboratories, this is a field where much improvement is needed.

It is shown that with appropriate care, a robust system can be designed and robustly solve navigational-dependant mission, regardless of what is literally thrown at it. This has been done using the JessMW package to add reasoning and validation in a non-intrusive way to a hard real-time capable system. The principle of using the expert system for robust mission control has also been proved by several inexperienced students in classes at Automation and Control, DTU Electrical Engineering.

REFERENCES

Andersen, J. C., 2006. *Mobile Robot Navigation,* Kgs Lyngby: Technical University of Denmark ØRSTED*DTU, Automation.

Arkin, R. c., 1989. *Towards the unification of navigational planning and reactive control.* s.l., Georgia Institute of Technology.

Beck, A. B., Andersen, N. A., Andersen, J. C. & Ravn, O., 2010. *MobotWare – A plug-in based framework for mobile robots.* s.l., IAV 2010. International Federation of Automatic Control.

Duan, Z.-h., Cai, z.-x. & Yu, J.-x., 2005. Fault diagnosis and fault tolerant control for wheeled mobile robots under unknown environments: A survey. *Proceedings of the 2005 IEEE International Conference on Robotics and Automation,* pp. 3428-3433.

Friedman-Hill, E., 1998. *Java expert system shell,* Sandia National Laboratories, Livermore, CA: s.n.

Grove, R., 2000. Internet-based expert systems. *Expert System,* July, 17(3), pp. 129-135.

Hladek, D., Vaščák, J. & Sinčák, P., 2009. Multi-robot control system for pursuit-evation problem. *Journal of Electrical Enfinerring,* 60(3), pp. 143-148.

Jørgensen, R. N., Nørremark, M., Sørensen, C. G. & Andersen, N. A., 2008. Utilising scripting language for unmanned and automated guided vehicles operating within row crops. *Computers and electronics in agriculture,* 62(2), pp. 190--203.

Marco, D. B., Healey, A. J. & McGhee, R. B., 1996. Autonomous underwater vehicles: Hybrid control of mission and motion. *Autonomous Robots,* 3(2-3), pp. 169-186.

Ong, S. K., An, N. & Nee, A. Y. C., 2001. Web-based fault diagnostic and learning system. *The International Journal of Advanced Manufacturing Technology,* 18(7), pp. 502-511.

Orchard, R., 2001. Fuzzy reasoning in Jess: The Fuzzy J toolkit and Fuzzy Jess. *Proceedings of the ICEIS 2001, Third International Conference on Enterprise Information Systems,* pp. 533-542.

Understrup, K. G., 2011. *Laserbaseret forhindringsundvigelse.* [Online]
Available at:
http://aut.elektro.dtu.dk/mobotware/doc/auavoidk.pdf

Zhou, F. et al., 2005. Control of an inspection robot for 110KV power transmission lines based on expert system design methods. *Control Applications, 2005. CCA 2005. Proceedings of 2005 IEEE Conference on ,* pp. 1563-1568.